Tipe data

C++

Atang Supena

Sejarah C++

Tahun 1978, Brian W. Kerninghan & Dennis M. Ritchie dari AT & T Laboratories mengembangkan bahasa B menjadi bahasa C. Bahasa B yang diciptakan oleh Ken Thompson sebenarnya merupakan pengembangan dari bahasa BCPL (Basic Combined Programming Language) yang diciptakan oleh Martin Richard.

Sejak tahun 1980, bahasa C banyak digunakan pemrogram di Eropa yang sebelumnya menggunakan bahasa B dan BCPL. Dalam perkembangannya, bahasa C menjadi bahasa paling populer diantara bahasa lainnya, seperti PASCAL, BASIC, FORTRAN.

Tahun 1989, dunia pemrograman C mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh American National Standards Institute (ANSI). Bahasa C yang diciptakan Kerninghan & Ritchie kemudian dikenal dengan nama ANSI C.

Mulai awal tahun 1980, Bjarne Stroustrup dari AT & T Bell Laboratories mulai mengembangkan bahasa C. Pada tahun 1983, lahirlah secara resmi bahasa baru hasil pengembangan C yang dikenal dengan nama C++. Bahasa ini bersifat kompatibel dengan bahasa pendahulunya yaitu C. Pada mulanya C++ disebut dengan "a better C". Nama C++ sendiri diberikan oleh Rick Mascitti pada musin panas 1983. Adapun tanda ++ berasal dari nama operator penaikan pada bahasa C.

Struktur Bahasa C++

Contoh1:

```
// my first program in C++
#include <iostream.h>
int main ()
{
  cout << "Hello World!";
  return 0;
}
```

Hasil:

```
Hello World!
```

Sisi kiri merupakan source code, yang dapat diberi nama hiworld.cpp dan sisi kanan adalah hasilnya setelah di-kompile dan di-eksekusi.

Program diatas merupakan salah satu program paling sederhana dalam C++, tetapi dalam program tersebut mengandung komponen dasar yang selalu ada pada setiap pemrograman C++. Jika dilihat satu persatu :

// my first program in C++

Baris ini adalah komentar. semua baris yang diawali dengan dua garis miring (//) akan dianggap sebagai komentar dan tidak akan berpengaruh terhadap program. Dapat digunakan oleh programmer untuk menyertakan penjelasan singkat atau observasi yang terkait dengan program tersebut.

#include <iostream.h>

Kalimat yang diawali dengan tanda (#) adalah preprocessor directive. Bukan merupakan baris kode yang dieksekusi, tetapi indikasi untuk kompiler. Dalam kasus ini kalimat #include <iostream.h> memberitahukan preprocessor kompiler untuk menyertakan header file standard iostream. File spesifik ini juga termasuk library deklarasi standard I/O pada C++ dan file ini disertakan karena fungsi-fungsinya akan digunakan nanti dalam program.

int main ()

Baris ini mencocokan pada awal dari deklarasi fungsi main. Fungsi main merupakan titik awal dimana seluruh program C++ akan mulai dieksekusi. Diletakan diawal, ditengah atau diakhir program, isi dari fungsi main akan selalu dieksekusi pertama kali. Pada dasarnya, seluruh program C++ memiliki fungsi main.

main diikuti oleh sepasang tanda kurung () karena merupakan fungsi. Pada C++, semua fungsi diikuti oleh sepasang tanda kurung () dimana, dapat berisi argumen didalamnya. Isi dari fungsi main selanjutnya akan mengikuti, berupa deklarasi formal dan dituliskan diantara kurung kurawal ({}), seperti dalam contoh.

cout << "Hello World";

Intruksi ini merupakan hal yang paling penting dalam program contoh. cout merupakan standard output stream dalam C++ (biasanya monitor). cout dideklarasikan dalam header file iostream.h, sehingga agar dapat digunakan maka file ini harus disertakan.

Perhatikan setiap kalimat diakhiri dengan tanda semicolon (;). Karakter ini menandakan akhir dari instruksi dan harus disertakan pada setiap akhir instruksi pada program C++ manapun. return 0;

Intruksi return menyebabkan fungsi main() berakhir dan mengembalikan kode yang mengikuti instruksi tersebut, dalam kasus ini 0. Ini merupakan cara yang paling sering digunakan untuk mengakhiri program.

Tidak semua baris pada program ini melakukan aksi. Ada baris yang hanya berisi komentar (diawali //), baris yang berisi instruksi untuk preprocessor kompiler (Yang diawali #),kemudian baris yang merupakan inisialisasi sebuah fungsi (dalam kasus ini, fungsi main) dan baris yang berisi instruksi (seperti, cout <<), baris yang terakhir ini disertakan dalam blok yang dibatasi oleh kurung kurawal ({}) dari fungsi main.

Struktur program dapat dituliskan dalam bentuk yang lain agar lebih mudah dibaca, contoh:

```
int main ()
{
  cout << " Hello World ";
  return 0;

  }
```

Atau dapat juga dituliskan :

```
int main () { cout << " Hello World "; return 0; }
```

Dalam satu baris dan memiliki arti yang sama dengan program-program sebelumnya. pada C++ pembatas antar instruksi ditandai dengan semicolon (;) pada setiap akhir instruksi.

Contoh2: **Hasil:**

```
// my second program in C++
#include <iostream.h>
int main ()
{
  cout << "Hallo! ";
  cout << "Selamat Datang di C++";
  return 0;
}
```

Hallo! Selamat Datang di C++

Komentar

Komentar adalah bagian dari program yang diabaikan oleh kompiler. Tidak melaksanakan aksi apapun. Mereka berguna untuk memungkinkan para programmer untuk memasukan catatan atau deskripsi tambahan mengenai program tersebut. C++ memiliki dua cara untuk menuliskan komentar :

```
//      Komentar baris
/*      Komentar Blok */
/*      Komentar seperti ini
        Juga biasa digunakan
        Di C++ */
```

Komentar baris, akan mengabaikan apapun mulai dari tanda (//) sampai akhir dari baris yang sama. Komentar Blok, akan mengabaikan apapun yang berada diantara tanda /* dan */.

Variabel, Tipe Data, Konstanta

Untuk dapat menulis program yang dapat membantu menjalankan tugas-tugas kita, kita harus mengenal konsep dari variabel. Sebagai ilustrasi, ingat 2 buah angka, angka pertama adalah 5 dan angka kedua adalah 2. Selanjutnya tambahkan 1 pada angka pertama kemudian hasilnya dikurangi angka kedua (dimana hasil akhirnya adalah 4).

Seluruh proses ini dapat diekspresikan dalam C++ dengan serangkaian instruksi sebagai berikut:

```
a = 5;
b = 2;
```

```
a = a + 1;
result = a - b;
```

Jelas ini merupakan satu contoh yang sangat sederhana karena kita hanya menggunakan 2 nilai integer yang kecil, tetapi komputer dapat menyimpan jutaan angka dalam waktu yang bersamaan dan dapat melakukan operasi matematika yang rumit.

Karena itu, kita dapat mendefinisikan variable sebagai bagian dari memory untuk menyimpan nilai yang telah ditentukan. Setiap variable memerlukan identifier yang dapat membedakannya dari variable yang lain, sebagai contoh dari kode diatas identifier variabelnya adalah a, b dan result, tetapi kita dapat membuat nama untuk variabel selama masih merupakan identifier yang benar.

Identifiers

Identifier adalah untaian satu atau lebih huruf, angka, atau garis bawah (_). Panjang dari identifier, tidak terbatas, walaupun untuk beberapa kompiler hanya 32 karakter pertama saja yang dibaca sebagai identifier (sisanya diabaikan). Identifier harus selalu diawali dengan huruf atau garis bawah (_).

Ketentuan lainnya yang harus diperhatikan dalam menentukan identifier adalah tidak boleh menggunakan keyword dari bahasa C++. Diawah ini adalah keyword dalam C++:

Asm	auto	bool	break	case
Catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
False	float	for	friend	goto
If	inline	int	long	mutable
namespace	new	operator	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t				

Sebagai tambahan, represetasi alternatif dari operator, tidak dapat digunakan sebagai identifier.

Contoh :

and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq

catatan: Bahasa C++ adalah bahasa yang "case sensitive", ini berarti identifier yang dituliskan dengan huruf kapital akan dianggap berbeda dengan identifier yang sama tetapi dituliskan dengan huruf kecil, sabagai contoh, variabel RESULT tidak sama dengan variable result ataupun variabel Result.

Tipe Data

Tipe data yang ada pada C++, berikut nilai kisaran yang dapat direpresentasikan :

TIPE DATA

Nama	Bytes	Description	Range*
*		character or integer 8 bits length.	signed: -128 to 127 unsigned: 0 to 255
char	1	integer 16 bits length.	signed: -32768 to 32767 unsigned: 0 to 65535
short long	2 4	integer 32 bits length.	signed:-2147483648 to 2147483647 unsigned: 0 to 4294967295
int	*	Integer. Its length traditionally depends on the length of the system's Word type, thus in MSDOS it is 16 bits long, whereas in 32 bit systems (like Windows 9x/2000/NT and systems that work under protected mode in x86 systems) it is 32 bits long (4 bytes).	See short, long
		floating point number.	3.4e + / - 38 (7 digits)
float	4		
double	8	double precision floating point number.	1.7e + / - 308 (15 digits)
long double	10	long double precision floating point number.	1.2e + / - 4932 (19 digits)
bool	1	Boolean value. It can take one of two values: true or false NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it. Consult section bool type for compatibility	true or false

		information.	
wchar_t	2	Wide character. It is designed as a type to store international characters of a two-byte character set. NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it.	wide characters

Deklarasi variabel

Untuk menggunakan variabel pada C++, kita harus mendeklarasikan tipe data yang akan digunakan. Sintaks penulisan deklarasi variabel adalah dengan menuliskan tipe data yang akan digunakan diikuti dengan identifier yang benar, contoh :

```
int a;
float mynumber;
```

Jika akan menggunakan tipe data yang sama untuk beberapa identifier maka dapata dituliskan dengan menggunakan tanda koma, contoh :

```
int a, b, c;
```

Tipe data integer (char, short, long dan int) dapat berupa signed atau unsigned tergantung dari kisaran nilai yang akan direpresentasikan. Dilakukan dengan menyertakan keyword signed atau unsigned sebelum tipe data, contoh :

```
unsigned short NumberOfSons;
signed int MyAccountBalance;
```

Jika tidak dituliskan, maka akan dianggap sebagai signed.

Contoh 3 :

```
// operating with variables        //
process:
#include <iostream.h>              a = 5;
int main ()                        b = 2;
{                                  a = a +
1;
      // declaring variables:      result =
 a - b;
 int a, b;
 int result;
```

Hasil

4

Inisialisasi Variabel

Ketika mendeklarasikan variabel local, kita dapat memberikan nilai tertentu. Sintaks penulisan sbb :

type identifier = initial_value ;

Misalkan kita akan mendeklarasikan variabel int dengan nama a yang bernilai 0, maka dapat dituliskan :

int a = 0;

Atau dengan cara lainnya, yaitu menyertakan nilai yang akan diberikan dalam tanda ():

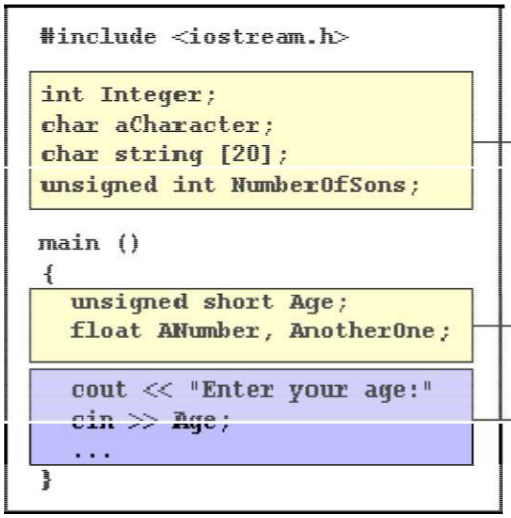
type identifier (initial_value) ;

Contoh :

int a (0);

Lingkup Variabel

Pada C++, kita dapat mendeklarasikan variable dibagian mana saja dari program, bahkan diantara 2 kalimat perintah.



variabel Global dapat digunakan untuk setiap bagian dari program, maupun fungsi, walaupun dideklarasikan diakhir program.

Lingkup dari variable local terbatas. Hanya berlaku dimana variable tersebut dideklarasikan. Jika dideklarasikan diawal fungsi (seperti dalam main) maka lingkup dari variable tersebut adalah untuk seluruh fungsi main. Seperti contoh diatas, jika terdapat fungsi lain yang ditambahkan pada main(), maka variable local yang dideklarasikan dalam main tidak dapat digunakan pada fungsi lainnya dan sebaliknya.

Pada C++, lingkup variable local ditandai dengan blok dimana variable tersebut dideklarasikan (blok tersebut adalah sekumpulan instruksi dalam kurung kurawal {}). Jika dideklarasikan dalam fungsi tersebut, maka akan berlaku sebagai variable dalam fungsi tersebut, jika dideklarasikan dalam sebuah perulangan, maka hanya berlaku dalam perulangan tersebut, dan seterusnya.

Konstanta : Literals.

Konstanta adalah ekspresi dengan nilai yang tetap. Terbagi dalam Nilai Integer, Nilai Floating-Point, Karakter and String.

Nilai Integer

Merupakan nilai konstanta numerik yang meng-identifikasikan nilai integer decimal. Karena merupakan nilai numeric, maka tidak memerlukan tanda kutip (") maupun karakter khusus lainnya. Contoh :

1776 707 -273

C++ memungkinkan kita untuk mempergunakan nilai oktal (base 8) dan heksadesimal (base 16). Jika menggunakan octal maka harus diawali dengan karakter 0 (karakter nol), dan untuk heksadesimal diawali dengan karakter 0x (nol, x). Contoh :

75 // decimal

0113 // octal

0x4b // hexadecimal

Dari contoh diatas, seluruhnya merepresentasikan nilai yang sama : 75.

Nilai Floating Point

Merepresentasikan nilai desimal dan/atau eksponen, termasuk titik desimal dan karakter e (Yang merepresentasikan "dikali 10 pangkat n" , dimana n merupakan nilai integer) atau keduanya.

Contoh :

3.14159 // 3.14159

6.02e23 // 6.02 x 10^{23}

1.6e-19 // 1.6 x 10^{-19}

3.0 // 3.0

Karakter dan String

Merupakan konstanta non-numerik, Contoh :

'z'

'p'

"Hello world"

"How do you do?"

Untuk karakter tunggal dituliskan diantara kutip tunggal (') dan untuk untaian beberapa karakter, dituliskan diantara kutip ganda (").
Konstanta karakter dan string memiliki beberapa hal khusus, seperti escape codes.

\n	Newline
\r	carriage return
\t	Tabulation
\v	vertical tabulation
\b	Backspace
\f	page feed
\a	alert (beep)
\'	single quotes (')
\"	double quotes (")
\?	question (?)
\\	inverted slash (\)

Contoh :

'\n'

'\t'

"Left \t Right"

"one\ntwo\nthree"

Sebagai tambahan, kita dapat menuliskan karakter apapun dengan menuliskan yang diikuti dengan kode ASCII, mengekspresikan sebagai octal (contoh, \23 atau \40) maupun heksadesimal (contoh, \x20 atau \x4A).

Konstanta Define (#define)

Kita dapat mendefinisikan sendiri nama untuk konstanta yang akan kita pergunakan, dengan menggunakan preprocessor directive #define. Dengan format :

#define identifier value

Contoh :

```
#define PI 3.14159265
#define NEWLINE '\n'
#define WIDTH 100
```

Setelah didefinisikan seperti diatas, maka kita dapat menggunakannya pada seluruh program yang kita buat, contoh :

```
circle = 2 * PI * r;
cout << NEWLINE;
```

Pada dasarnya, yang dilakukan oleh kompiler ketika membaca #define adalah menggantikan literal yang ada (dalam contoh, PI, NEWLINE atau WIDTH) dengan nilai yang telah ditetapkan (3.14159265, '\n' dan 100). #define bukan merupakan instruksi, oleh sebab itu tidak diakhiri dengan tanda semicolon (;).

Deklarasi Konstanta (const)

Dengan prefix const kita dapat mendeklarasikan konstanta dengan tipe yang spesifik seperti yang kita inginkan. contoh :

```
const int width = 100;
const char tab = '\t';
const zip = 12440;
```

Jika tipe data tidak disebutkan, maka kompiler akan meng-asumsikan sebagai int.

Operator

Operator-operator yang disediakan C++ berupa keyword atau karakter khusus. Operator-operator ini cukup penting untuk diketahui karena merupakan salah satu dasar bahasa C++.

Assignation (=).

Operator assignation digunakan untuk memberikan nilai ke suatu variable.

```
a = 5;
```

Memberikan nilai integer 5 ke variabel a. Sisi kiri dari operator disebut lvalue (left value) dan sisi kanan disebut rvalue (right value). lvalue harus selalu berupa variabeldan sisi kanan dapat berupa konstanta, variabel, hasil dari suatu operasi atau kombinasi dari semuanya.

Contoh :

```
int a, b;  // a:? b:?
a = 10;    // a:10 b:?
b = 4;     // a:10 b:4
a = b;     // a:4 b:4
b = 7;     // a:4 b:7
```

Hasil dari contoh diatas, a bernilai 4 dan b bernilai 7.

Contoh :

```
a = 2 + (b = 5);
```

equivalen dengan :

```
b = 5;
a = 2 + b;
```

Arithmetic operators (+, -, *, /, %)

+ addition

- subtraction

*** multiplication**

/ division

% module

Compound assignation operators

(+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

contoh :

value += increase; equivalen dengan value = value + increase; a -= 5; equivalen dengan a = a - 5; a /= b; equivalen dengan a = a / b; price *= units + 1; equivalen dengan price = price * (units + 1);

Increase (++) and decrease (--).

Contoh :

```
a++;
a+=1;
a=a+1;
```

Contoh diatas adalah equivalen secara fungsional. Nilai a dikurangi 1.

Operator Increase dan Decrease dapat digunakan sebagai prefix atau suffix. Dengan kata lain dapat dituliskan sebelum identifier variabel (++a) atau sesudahnya (a++). operator increase yang digunakan sebagai prefix (++a), Perbedaannya terlihat pada tabel dibawah ini :

Example 1

```
B=3;
A=++B;
// A is 4, B is 4
```

Example 2

```
B=3;
A=B++;
// A is 3, B is 4
```

Pada contoh 1, B ditambahkan sebelum nilainya diberikan ke A. Sedangkan contoh 2, Nilai B diberikan terlebih dahulu ke A dan B ditambahkan kemudian.

Relational operators (==, !=, >, <, >=, <=)

Untuk mengevaluasi antara 2 ekspresi, dapat digunakan operator Relasional. Hasil dari operator ini adalah nilai bool yaitu hanya berupa true atau false, atau dapat juga dalam nilai int, 0 untuk merepresentasikan "false" dan 1 untuk merepresentasikan "true". Operator-operator relasional pada C++ :

== Equal

!= Different

> Greater than

< Less than

>= Greater or equal than

<= Less or equal than

Contoh :

(7 == 5) would return false.

(5 > 4) would return true.

(3 != 2) would return true.

(6 >= 6) would return true.

(5 < 5) would return false.

Contoh, misalkan a=2, b=3 dan c=6 :

(a == 5)	would return	false.
(a*b >= c)	would return	true since (2*3 >= 6) is it.
(b+4 > a*c)	would return	false since (3+4 > 2*6) is it.
((b=2) == a)	would return	true.

Logic operators (!, &&, ||).

Operator ! equivalen dengan operasi boolean NOT, hanya mempunyai 1 operand, berguna untuk membalikkan nilai dari operand yang bersangkutan. Contoh :

!(5 == 5)	returns false because the expression at its right (5 == 5) would be true.
!(6 <= 4)	returns true because (6 <= 4) would be false.
!true	returns false.
!false	returns true.

operator Logika && dan || digunakan untuk mengevaluasi 2 ekspresi dan menghasilkan 1 nilai akhir. mempunyai arti yang sama dengan operator logika Boolean AND dan OR. Contoh :

First Operand a	Second Operand b	result a && b	result a || b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Contoh :

((5 == 5) && (3 > 6)) returns false (true && false).

((5 == 5) || (3 > 6)) returns true (true || false).

Conditional operator (?).

operator kondisional mengevaluasi ekspresi dan memberikan hasil tergantung dari hasil evaluasi (true atau false). Sintaks :

condition ? result1 : result2

Jika kondisi true maka akan menghasilkan result1, jika tidak akan menghasilkan result2.

7==5 ? 4 : 3 **returns 3 since 7 is not equal to 5.**

7==5+2 ? 4 : 3 returns 4 since 7 is equal to 5+2.

5>3 ? a : b **returns a, since 5 is greater than 3.**

a>b ? a : b **returns the greater one, a or b.**

Bitwise Operators (&, |, ^, ~, <<, >>).

Operator Bitwise memodifikasi variabel menurut bit yang merepresentasikan nilai yang disimpan, atau dengan kata lain dalam representasi binary.

op	asm	Description
&	AND	Logical AND
|	OR	Logical OR
^	XOR	Logical exclusive OR
~	NOT	Complement to one (bit inversion)
<<	SHL	Shift Left

>>	SHR	Shift Right

Explicit type casting operators

Type casting operators memungkinkan untuk mengkonversikan tipe data yang sudah diberikan ke tipe data yang lain. Ada beberapa cara yang dapat dilakukan dalam C++, yang paling popular yaitu tipe baru dituliskan dalam tanda kurung () contoh:

```
int i;
float f = 3.14;
i = (int) f;
```

Contoh diatas, mengkonversikan nilai 3.14 menjadi nilai integer (3). Type casting operator yang digunakan (int). Cara lainnya :

```
i = int ( f );
```

sizeof()

Operator ini menerma 1 parameter, dapat berupa type variabel atau variabel itu sendiri dan mengembalikan ukurannya type atau object tersebut dalam bytes :

```
a = sizeof (char);
```

Contoh diatas akan memberikan nilai 1ke a karena char adalah tipe data dengan panjang 1 byte. Nilai yang diberikan oleh sizeof bersifat konstsn constant.

Prioritas pada operator

Contoh :

a = 5 + 7 % 2

Jawaban dari contoh diatas adalah 6. Dibawah ini adalah prioritas operator dari tinggi ke rendah :

Priority Operator	Operator	Description
1	::	Scope
2	()[]->	Sizeof
3	++ --	increment/decrement
	~	Complement to one (bitwise)
	!	unary NOT
	& *	Reference and Dereference (pointers)
	(type)	Type casting
	+ -	Unary less sign
4	* / %	arithmetical operations
5	+ -	arithmetical operations
6	<< >>	bit shifting (bitwise)
7	<<=>>=	Relational operators
8	== !=	Relational operators
9	& ^ |	Bitwise operators
10	&& ||	Logic operators
11	?:	Conditional
12	=+=-=*=/=%=	Assignation
	>>= <<= &= ^= |=	
13	,	Comma, Separator

Daftar Pustaka

1. **Deitel & Deitel, C How to Program 3rd Edition, Prentice Hall, New Jersey, 2001**
2. **Jogiyanto, Konsep Dasar Pemrograman Bahasa C, Andi Offset, Yogyakarta, 1993**
3. **Moh. Sjukani, 2009, Algoritma & Struktur Data 1 dengan C, C++ dan Java, Edisi 5, Mitra Wacana Media, Jakarta.**
4. **Thompson Susabda Ngoen, Pengantar Algoritma dengan Bahasa C, Salemba Teknika, Jakarta, 2004.**